COMBINATION OF IDDQ TESTING AND HIGH LEVEL ATPG

András Petri
Budapest University of Techology and Economic Sciences
Department of Measurement and Information Systems

Abstract: *Test generation for today's complex digital circuits is an extremely computation intensive task. The search space of ATPG can be reduced by starting from higher level circuit descriptions. The integration of alternate testing methodology - I_{DDQ} testing - is suggested for increasing the efficiency of a high level VHDL based test generator.*

1. Introduction

The ever growing complexity of digital circuits have imposed greater and greater demand on test pattern generation algorithms for the past twenty years. Though promising solutions have been elaborated to speed up the test generation process, no algorithm can cope with its excessive computational complexity, as it is proven that the ATPG problem is NP-complete. An attractive path for the circumvention of this obstacle can be the *raise of the abstraction level* of the input of the ATPG procedure, in order to reduce the structural complexity of the circuit.
A high level automatic test pattern generation tool [4] was developed within the frame of the FUTEG [1] project. The tool accepts architectural level digital circuit descriptions in VHDL language, and produces functional level test vectors for a given fault set. The test generation is executed by applying constraint satisfaction techniques, and is enhanced by heuristics-based constraint preprocessing and scheduling methods.
The conventional circuit testing paradigm provides fault detection by applying a test vector stimulus to the inputs, and observing the outputs of the circuits. An alternate paradigm is applied in the case of *I_{DDQ} testing* [5]. This technique observes the effect of the faults activated by test stimuli at the power supply line of the circuit, thus providing an opportunity of simplification in automatic test generation algorithms.
Incorporation of I_{DDQ} testing methodology into the BudaTest ATPG system is a promising idea. Due to the nature of its basic test generation principles, inherited from traditional gate level algorithms, a significant gain is expected in test generation efficiency. A preliminary experimental version of the I_{DDQ}-enhanced BudaTest tool is under development.

2. The BudaTest ATPG Tool

BudaTest is an automatic test generator for digital circuits described on architectural functional level in VHDL language, thus it was proposed to be used typically after the scheduling and resource allocation phases but, it does not assume that the logic was already unfolded into a sea of gates.The tool applies systematic search methods in order to find test vectors for a given set of faults. It is a modular prrogtam implemented in C++ language.

It must be pointed out that one of the main design guidelines of BudaTest was *direct support of the early design process*, in terms that the VHDL subset accepted by BudaTest is identical to the synthetizable subset provided by the AMICAL [7] behavioural level synthesis tool. This way the user has a complete chain of technologies covering the behavioural to structural transformation and the test and testability analysis as well.

2.1. The fault model

The validity of the fault model is one of the most important issues considered at the selection of the ATPG model. On the functional architectural level, VHDL language constructs have direct counterparts in the term of hardware components and signals; hence, physical faults affecting interconnections are manifested as storage problems of these signals. The traditional *stuck-at* and *short* fault model is applicable here, too; moreover, the model used by BudaTest makes the handling of multiple bit-faults on a single high-level signal possible.
Faults of the functional units (FUs) can be represented as purely functional faults, e.g. the execution of a wrong

operation. However, in an ATPG tool integrated into a design system based on FU libraries, a more accurate fault representation is possible with the use of either

- *faulty unit libraries*, where library elements are obtained by the back-annotation of lower level faults, or
- *test scenarios*, where test patterns for FUs are given together with the FUs themselves. Test scenarios are actually high level algorithmic extensions of the traditional *primitive cube of fault* notation.

Both fault representations can be used for the constraint based approach. The test scenario representation provides a very useful feature of *hierarchical test generation*: the test patterns of a simple component can be reused when generating test vectors for a more complex circuit. This feature allows the easy integration of I_{DDQ} test vectors into the BudaTest tool.

3. I_{DDQ} Testing

The traditional digital circuit testing methodology is based on *voltage measurement*; i.e. the observation of the voltage level, as a logical value, appearing on a circuit output while the appropriate logical values are forced to the circuit's inputs. However, many of the physical, transistor level faults in digital circuits can be detected by completely different, *current measurement* methods. I_{DDQ} testing is one of the most promising and most well-elaborated of these current based techniques [3].

3.1. Principle of Fault Detection

Most of the high-complexity digital circuits used by today's electronics industry are built with CMOS technology. The basic scheme of all CMOS circuits consists of two blocks: a network of p-channel MOSFETs between the power supply line (V_{DD}) and the output, and a network of n-channel MOSFETs between the ground (V_{SS}) and the output. Due to the properties of MOSFETs, I_{DD}, the current flowing on the V_{DD} power supply line is usually very low, expect during the time of switching transients. The value of I_{DD} in a *stabilized, quiescent* state of the circuit, noted as I_{DDQ}, is typically in the magnitude of femtoamperes (10^{-15} A) in the case of a single transistor pair, and still in the magnitude of nanoamperes for a whole complex CMOS circuit.
The most common class of the physical faults in CMOS circuits are the *leakage faults*, appearing as shorts or conductances between various points of the circuit. These parasitic conductances increase I_{DDQ} with a few orders of magnitude. This phenomenon makes the detection of physical faults possible by measuring I_{DDQ} while 'activating' the fault in question; that is, ensuring that current is flowing through the faulty conductance.
As the quiescent current of CMOS circuits is extremely low, its measurement is possible only by indirect techniques. The most common method [6] involves applying power onto the circuit under test then removing it abruptly. After then, the time function of the voltage is recorded on the V_{DD} pin.
The circuit is considered as an *RC* couple in which *R* represents the overall isolation resistance of the circuit (including shorts and parasitic conductances) and *C* represents the capacitance between the circuit's power supply lines (including any external capacitors attached). If the circuit has I_{DDQ}-detectable faults, the time constant of this *RC* couple is significantly shorter than in the case of a fault-free circuit.

3.2. Benefits of I_{DDQ} Testing

The I_{DDQ} based fault detection methodology has many significant and attractive advantages over the traditional value based methods:

- *Test vector generation* for I_{DDQ} testing is *much simpler* and more efficient. As the observation of the test vector's effect is made always on the power supply line, the effect of an activated fault is *not necessary to propagate* towards the outputs. This feature results in a drastic reduction of the search space in systematic test generation algorithms.
- The *range of considered faults* is significantly *wider*. Additionally to the classical *stuck-at* faults, I_{DDQ} testing is able to detect the following fault types as well:
    - faults related to *semiconductor manufacturing technology*: gate oxide defects, parasitic junctions, shorts between the transistors' terminals, etc.;
    - *hidden defects* that does not affect the operation of the circuit under normal circumstances, but de-

crease its reliability under extreme conditions. These faults have special importance as they are completely undetectable by logic value based methods, but their detection and elimination is essential in high reliability circuits.

- *Multiple faults* can be detected simultaneously by a *single test vector.* Contrary to the traditional methods, when the propagation and observability requirements of different faults may be contradictory, each I_{DDQ} test vector will detect *every fault* that was activated by it, as the effects of all activated faults in the circuit is summed up.

These advantages make I_{DDQ} testing applicable in many areas where value based testing is difficult or even impossible, like silicon prototype testing and circuit reliability analysis.

4. Integration of I_{DDQ} Techniques into BudaTest

Although the I_{DDQ} testing methodology is seemingly closely related to the lowest level of digital circuit descriptions, while the BudaTest tool operates on a much higher level of abstraction, it is a reasonable idea to combine them in order to enhance the test generation capabilities, as many features of I_{DDQ} testing fit well to the structure of BudaTest:

- The faults detectable by I_{DDQ} methods are either considered in the BudaTest ATPG tool, or strongly related to the physical level manufacturing technology (e.g. gate oxide faults, hidden defects) and thus they have no practical importance from the point of view of high level test generation.
- As I_{DDQ} testing applies the same fault sensitization technique as the classical methods, all value based test vectors generated by BudaTest can be considered as I_{DDQ} test vectors as well.
- Pre-calculated I_{DDQ} test vectors of individual components, generated by low level TPG systems or supplied by the component manufacturers, can be exploited by BudaTest when generating tests for complex circuits, as they can be considered as local test scenarios (see Section 2.1).

Due to the structure of the constraint based test generator of BudaTest and the carefully designed C++ object oriented implementation, incorporation of new test methodologies is rather simple, and requires usually only minor code modifications.

4.1. BudaTest as an I_{DDQ} ATPG

The incorporation of I_{DDQ} capabilities is feasible in many different parts of the BudaTest system. First of all, the original value based test generator can be used in a very straightforward way for generating I_{DDQ} test vectors as well, without modifying the core ATPG engine itself! This feature can be implemented by applying an *alternate node labelling* algorithm during the circuit description preprocessing phase.

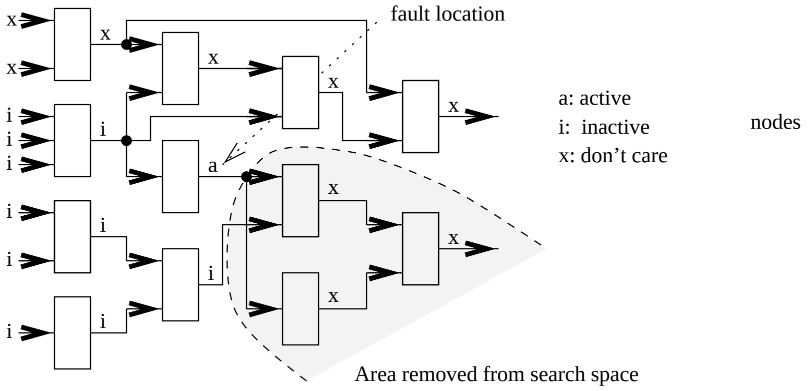

Fig. 1. Node labelling for I_{DDQ} testing

The node labelling for I_{DDQ} test generation (Figure 2) is a simplified version of the one described in [2]. As fault propagation towards the circuit's outputs and thus the distinction of 'potentially active' nodes is unnecessary in this case, the selection of an 'active' output node is omitted, and 'potentially active' nodes are qualified as 'dont' care' instead.This simple change enhances the speed of the test generation algorithm dramatically, and it can be implemented as an additional option of the ATPG engine while all the appropriate existing heuristics remain applicable.

4.2. I_{DDQ} Based Algorithmic Enhancements

As the I_{DDQ} test techniques have certain discouraging features besides its advantages, they are often applied *in a combination with* traditional *value based* testing to exploit the best from both methodologies. Similarly, the BudaTest system can be used not only for I_{DDQ} test vector generation, but as a general framework for various combinations of value based and I_{DDQ} based testing methods.

So far, the following I_{DDQ} based *algorithm enhancements* have been elaborated, and partially implemented within BudaTest:

- As it was mentioned in the beginning of Section 4, every value based test vector of BudaTest is also an I_{DDQ} test vector for the same fault. Moreover, every test vector potentially activates many other I_{DDQ}-detectable faults as well.
- The number of activated I_{DDQ} faults can be used as a *heuristic measure for decision support* [1] and for qualification of test vectors, as the more faults are activated by a single test vector, the greater is the quiescent current in a faulty circuit, and thus the easier is the fault detection. Therefore those decisions should be preferred that activate the maximal number of I_{DDQ} faults.
- The number of detected I_{DDQ} faults can also be supplied to the end user for use outside of the BudaT est system, for *scheduling and organization the circuit test* process. Due to the extensive time requirements of a single I_{DDQ} measurement, the application of *selective I_{DDQ} testing* [6] is a common practice that consists of running a value based test set at normal operation speed, "paused" at certain test vectors in order to perform I_{DDQ} measurements. The most suitable test vectors can be chosen by their I_{DDQ} detection capability.

In order to enhance the effectiveness of testing, it is practical to *combine* individual test vectors, generated for single faults, into *compact test vectors* that are able to detect multiple faults simultaneously. Two test vectors can be combined only if they are *compatible*, that is, they do not contain contradictory value assignments and do not imply value contradictions in the circuit when applied simultaneously. Compaction of the generated test vector set can be done by determining the *maximal compatible clusters* of the vectors. I_{DDQ} test vectors confine significantly smaller parts of the circuit than value based ones, and thus the probability of incompatibility between any two vectors is much lower. Moreover, due to the cumulative effect of activated I_{DDQ} faults, compact I_{DDQ} test vectors have better fault detection capability.

References

[1] B. Sallay, K. Tilly, A. Pataricza, Z. Hegedüs, A. Petri, L. Surján, J. Sziray: *An Artificial Intelligence Based Approach to VHDL-Level Test Pattern Generation.* Technical Report FUTEG-4/1994 (FUTEG PECO Project 9624), 1995, Budapest.

[2] B. Sallay, A. Petri, A. Pataricza, K. Tilly: *BudaTest: An architectural level heuristics-driven test generation tool. Description of the prototype.* Technical Report FUTEG-8/1996 (FUTEG PECO Project 9624), 1996, Budapest.

[3] E. Gramatova, J. Gaspar, H. Manhaeve: *Test Pattern Generation for IDDQ/Voltage Testing Based on Fault Simulator for Combinational Circuits.* Proc. of DDECS'98 Workshop, September 1998, Szczyrk, Poland.

[4] B. Sallay, A. Petri, K. Tilly, A. Pataricza, J. Sziray: *High Level Test Pattern Generation for VHDL Circuits.* Proceedings of the IEEE European Test Workshop '96, June 1996, Montpellier, pp. 201-205.

[5] Y. K. Malaiya, S. Y. H. Su: *A New Fault Model and Testing Technique for CMOS Devices.* ITC, pp. 25-34, November 1982.

[6] J. Soden, C. F. Hawkins, R, K, Gulati, W. Mao: *IDDQ Testing: A Review*, Journal of Electronic Testing: Theory and Applications, 1992, pp. 5-17.

[7] A. A. Jerraya et al: *AMICAL - Interactive Architectural Synthesis Based on VHDL.* INPG/TIMA System Level Synthesis Group, Grenoble, 1994.